SUBJECT: SOFTWARE DESIGN 2	ANNEXURE(S) Formulas	PAGES 3	TIME 3 HRS
CODE: SDN260S	DATE 15 August 2022	MARKS 75	

Cape Peninsula University of Technology

FACULTY OF ENGINEERING

ASSESSMENT I: SEMESTER TWO MEMO

COURSE: BET: COMPUTER ENGINEERING

EXAMINER	:	Haltor Mataifa
MODERATOR (INTERNAL)	:	Mr. V. Moyo
MODERATOR (EXTERNAL)	:	N/A

SPECIAL INSTRUCTIONS

1. Answer all questions
2. Write your name and student number on each page of everything you submit
3. Write answers to theoretical questions in an MS Word document
4. Create a new project for each programming problem
5. Zip all files and folders together and submit as one zipped file on Blackboard
6. Be sure to add comments to your program code to make it understandable
7. You have a choice to answer **either question 2 or question 3** (**not both**). **All other questions must all be answered** (i.e. questions 1 and 4)

QUESTION 1 **[23]**

1.1. Explain the main difference between a String and a StringBuilder. Also state when it would be advisable to use the one instead of the other [**2**]

***Answer**:*

- *A String is **immutable** in Java (meaning it cannot be modified once instantiated), whereas a StringBuilder is **mutable** (i.e. its value can be changed once instantiated).*
- *It is preferable to use StringBuilder where the value of a string is expected to change over its lifetime, and to use String when the value of the string object is not required to change over its lifetime*

1.2. Explain the difference between a String literal and a String object instantiated using the new operator in Java [2]

***Answer**:*

- *Java treats **String literals with the same content** as a **single String object** with many references to it, whereas when two String objects are instantiated using the **new** operator, they will be treated as **two different objects**, even if they may have the same content*

1.3. Explain the difference between string methods equals and compareTo [2]

***Answer**:*

- *Both **equals** and **compareTo** String methods check the equality of the contents of two strings; the main difference is that **equals returns a boolean type** (true when contents are the same, false otherwise), whereas **compareTo returns an integer type** (which may be less than zero, equal to zero or greater than zero, depending on how the contents of the two string compare)*

1.4. Consider the following piece of code:

```
1  String s1=new String("Welcome to SDN260S");
2  String s2=" welcome to SDN260S ";
3. String s3=new String("Welcome to SDN260S");
4. s3==s1;
5. String s4=s2;
6. s4==s2;
```

1.4.1 What is the purpose of the statement in line 4? What is the result after executing this line?

[**3**]

***Answer**:*

- *Line 4 **checks whether s3 and s1 are references pointing to the same String object**. The result after execution of this line is "**false**" (because **s3** and **s1** do not refer to the same object in memory)*

1.4.2 What is the purpose of the statement in line 5? What is the result after executing this line?

[**3**]

Answer:

Line 5 instantiates String object s4 and assigns to it the same value as that of s2. The result after execution of this line is that the value of s4 is "welcome to SDN260S"

1.4.3 What is the purpose of the statement in line 6? What is the result after executing this line?

[**3**]

Answer:

Line 6 ***checks whether s4 and s2 are references pointing to the same String object****. The result after execution of this line is "****true****" (because s4 and s2 do point to the same string literal)*

1.5. What is a *recursive method*? How does it differ from a *standard method*? [2]

Answer*:*

- *A recursive method is a method that calls itself within the body of the method. It differs from standard method in its structure, because it always has two cases: the base case and the recursive method call. The base case must eventually be reached for the recursive method to terminate successfully*

1.6. Compare recursion with iteration. What are their similarities and differences? When is it preferable to use recursion, and when should it be avoided? [6]

Answer*:*

- *Main similarity between recursion and iteration is that they both make use of a control statement, repetition, and a termination test*
- *The differences lie in the way these three components are implemented (e.g. iteration uses repetition statement such as for…, while…, whereas recursion uses selection such as if…then…else)*
- *Recursion can be used when it more naturally mirrors the problem to be solved, because then it is relatively easier to code. It should be avoided when high performance (e.g. speed, memory) is of concern, because recursion tends to be computation-intensive*

QUESTION 2: **[30]**

Write a Java application that will request a user to enter a sentence, after which it will perform the following operations on the sentence:

- Firstly, it will split the sentence into individual words; assume that the words of the sentence are separated by white space
- It will then output the words to the screen, each word on a separate line
- The first letter of each word must be capitalized before printing to the screen

- The words will also be written to a text file (each on a separate line) in the order that they are printed on the screen. Be sure that writing subsequent words to the text file does not result in overwriting the previously written words

Answer:

Source code:

```
// Question 2:
    // Instantiate a (character-based) input stream object (Scanner):
    Scanner input = new Scanner(System.in);
    System.out.println("Please enter a sentence: ");
    String sentence=input.nextLine();

    // Split the sentence into individual words:
    String[] words=sentence.split(" ");

    // Instantiate (character-based) output stream object:
    Formatter outputWriter=null;

    try{
       outputWriter = new Formatter(new BufferedWriter(new FileWriter("words.txt", true)));
    }
    catch (FileNotFoundException fnfe){
       System.err.println("Error opening or creating file.");
       System.exit(1);
    }
    catch (IOException ioe){
       System.err.println("Error writing to file.");
       return;
    }

    // Print output to screen:
    System.out.println("\nThank you. Following is what you entered:");
    for (String word: words){
       StringBuilder s=new StringBuilder(word);
       s.setCharAt(0, Character.toUpperCase(word.charAt(0)));
       System.out.printf("%s\n", s.toString());
       outputWriter.format("%s\n", s.toString());
    }
    if (outputWriter != null)
       outputWriter.close();

    if(input != null)
       input.close();
```

QUESTION 3: **[30]**

Write a Java application that will validate a telephone number entered by a user. The application should check the validity of the telephone number entered against the following requirements:

i. The format should be (XXX) XXX-XXXX (i.e. ten digits)

ii. The first three digits, enclosed in parentheses, are the area code

iii. The next three digits should be separated from the area code by single white space, and by a hyphen from the last four digits

iv. The area code should always begin with 0

v. The second digit of the area code should lie between 0 and 5

vi. The remaining (eight) digits can lie anywhere between 0 and 9

vii. If the user enters an invalid telephone number, the application should inform the user and grant them two more attempts to enter a telephone number in the correct format

viii. After three unsuccessful attempts, the application should output an appropriate message to the user and terminate the program

ix. Once the user has entered a telephone number in the required format, the application should thank the user and print the entered telephone number to the screen in the following format (on separate lines):

Area code: XXX

Phone number XXX-XXXX

x. The application needs to *show the user the correct format in which the telephone number needs to be entered* (although it need not inform the user about the limits on the digits as specified in (iv) – (vi))

Answer:

Source code:

```
// Question 3:
    // Instantiate a (character-based) input stream object (Scanner):
    Scanner input = new Scanner(System.in);

    // Request user to enter telephone number:
    System.out.println("Please enter telephone number in the format: (XXX) XXX-XXXX\n"
        + "3-digit code must be enclosed in parentheses\n"
        + "with space between area code and next 3 digits\n"
        + "and hyphen between next 3 digits and last 4 digits: ");
    String telNumber=input.nextLine();

    // Formulate regular expression:
    String regexp="\\(0[0-5][0-9]\\) [0-9]{3}-[0-9]{4}";

    boolean telNumberMatches=telNumber.matches(regexp);

    int i=1;

    while (!telNumberMatches && i<3){
       System.out.println("\nEntry is not valid.");
       System.out.println("\nPlease enter telephone number in the format: (XXX) XXX-XXXX\n"
          + "3-digit code must be enclosed in parentheses\n"
          + "with space between area code and next 3 digits\n"
          + "and hyphen between next 3 digits and last 4 digits: ");
       telNumber=input.nextLine();
       telNumberMatches=telNumber.matches(regexp);
       i+=1;
    }
```

```
if (!telNumberMatches)
   System.out.println("\nMaximum number of attempts reached. Thank you and goodbye.");
else {
   String[] telNumberParts=telNumber.split(" ");
   String areaCode=telNumberParts[0].substring(1, 4);
   System.out.printf("\nThank you. Following is the information you entered:\n"
         + "Area code: %s\nPhone number: %s", areaCode, telNumberParts[1]);
}
```

QUESTION 4: **[22]**

Write a Java application that will use a recursive method to *add all the odd numbers from 0 up to the* (positive integer) *value entered by the user*. The application will do the following:

- Request the user to enter the (positive integer) number which is the upper limit for the computation. For example, if the user enters number=10, the application will add all odd numbers between 0 and 10
- Repeatedly prompt the user until they enter a positive number (in case the user entered a number that is zero or negative)
- Use the recursive method for computing the sum of odd numbers
- Output the result to the screen as "sum of odd numbers between 0 and n = result", where n is the number entered by the user, and result is the result obtained by applying the recursive method

Answer:

Source code:

```
// Question 4:
        // Instantiate a (character-based) input stream object (Scanner):
        Scanner input = new Scanner(System.in);

        // Request user to enter upper limit for the computation:
        System.out.println("Please enter maximum number up to which "
              + "sum of odd numbers must be computed: ");

        int n=input.nextInt();

        while (n<=0){
           System.out.println("\nNumber must be greater than zero.");
           System.out.println("Please enter maximum number up to which "
              + "sum of odd numbers must be computed: ");

        n=input.nextInt();
        }

        System.out.printf("sum of odd numbers between 0 and %d is %d\n",
              n, recursiveSumOddNumbers(n));
        }

     // Define recursive method to compute sum of odd numbers up to n:
```

```
public static int recursiveSumOddNumbers(int n){
   if (n%2==0)
      n-=1;

   if (n==1)
      return n;
   else
      return n+recursiveSumOddNumbers(n-2);
}
```

End of Assessment